MICAz 上での η_T ペアリング高速実装

北村 晃輔 †　　白勢 政明 †

† 公立はこだて未来大学情報アーキテクチャ学科
〒 041-8655 北海道函館市亀田中野町 116 番地 2

あらまし 近年, MICAz のマイクロコントローラである ATmega128L 上での公開鍵暗号の高速実装が盛んに行われている. 例えば, ペアリング暗号の実装として, Oliverira 等による TinyPBC, 石黒等による η_T ペアリング実装, Szczechowiak 等による NanoECC, 宮崎等による η_T ペアリング実装などがあげられる. 本稿では, MICAz 上に標数 3 を用いた η_T ペアリングの実装を, 拡大次数 $m = 97$ を用いて行った. 初期実装では nesC 言語を用いて実装を行い, ペアリングの演算時間は 5.2 秒であった. その後, 川原等によって提案された最小論理命令数での有限体の加算を用い, 有限体の加算及び乗算の高速化を行った. その結果, ペアリングの演算時間は 3.9 秒となり, 石黒等の標数 3 を用いた η_T ペアリングより 1.5 倍の高速化を達成した.

Efficient Implementation of the η_T Pairing on MICAz

Kosuke KITAMURA†　　Masaaki SHIRASE†

†Future University Hakodate, School of Systems Information Science
116-2, Kamedanakano-cho, Hakodate, Hokkaido, 041-8655, Japan

Abstract Recently, several implementations of public-key cryptography on ATmega128L that is CPU of MICAz have been reported. For implementations of Pairing Based Cryptosystems, TinyPBC by Oliveira et al., η_T pairing by Ishiguro et al., NanoECC by Szczechowiak et al., and η_T pairing by Miyazaki et al., have been reported. In this paper, we present an efficient implementation of the η_T pairing over finite fields of characteristic 3 with extension degree $m = 97$ on MICAz. The execution time of the pairing was 5.2 seconds in our initial implementation using nesC language. We then enhanced the speed of addition and multiplication over the finite field using minimum number of logical instructions for addition proposed by Kawahara et al. As a result, execution time for the η_T pairing becomes 3.9 seconds. We achieved 1.5 times speed-up from an the η_T pairing over the finite field of characteristic 3 by Ishiguro et al.

1　はじめに

近年,「いつでも, どこでも, 誰でも」使えるユビキタス社会を, セキュリティ確保やプライバシー保護等に留意して, 実現することが求められている [9]. また, ユビキタス端末等において瞬時に安全かつ確実に認証を行う技術や, 相手に応じて適切な情報のみを提供可能とするプライバシー保護技術の実現が求められている [10]. このように, ユビキタス端末においての安全かつ高度な利用, 活用が求めれている.

本稿では, ユビキタス社会を実現するインフラの一つである, 無線センサネットワークに注目する. 無線センサネットワークは何らかのセンサを持つ端末を測定対象となる領域あるいは人, 物に設置し, ネットワークを構成することで, 対象の状況, 情報などを収集するものである. 無線センサネットワークでもセキュリティの問題が挙げられている. Perrig 等は, 複数のノード同士の鍵共有, 機密性と認証, プライバシーの保守等の問題を指摘している [18]. これらの解決手段の一つとして公開鍵暗号方式が適している. しかしながら, 無線センサネットワークを構築するセンサノードは, CPU の周波数やメモリ容量などの計算資源が乏しく, 処理性能が低い. そのため, Perrig 等はセンサノード上での公開鍵暗号の実装が困難であると指摘している. 従って, センサノード上での公開鍵暗号の実装には, センサノードに特化した高速化を行う必要がある.

MICAz は世界中で研究用センサノード端末として最も多く利用されている [13]. また, MICAz のマイクロコントローラである ATmega128L 上では様々な公開鍵暗号の実装が盛んに行われている. ペアリング暗号の実装として, Oliverira 等により TinyTate[16] と TinyPBC[17], 石黒等による実装 [8], Szczechowiak

等による NanoECC[20], 宮崎等による実装 [14] などがあげられる.

TinyTate は $\mathbb{F}_q$(q は 512 ビットの素数) 上の埋め込み次数 2 の Tate ペアリング実装であり, 演算時間 30.21 秒という結果を示した. TinyPBC は有限体 $\mathbb{F}_{2^{271}}$ 上の埋め込み次数 4 の η_T ペアリング実装であり, 演算時間 5.45 秒という結果を示した. 石黒等の実装は有限体 $\mathbb{F}_{3^{97}}$ 上の埋め込み次数 6 の η_T ペアリングを用い, 演算時間 5.79 秒という結果を示した. NanoECC は有限体 $\mathbb{F}_{2^{271}}$ 上の埋め込み次数 4 の η_T ペアリング実装と, 有限体 $\mathbb{F}_p$(p は 256 ビットの素数) 上の埋め込み次数 4 の Ate ペアリング実装である. 演算時間は η_T ペアリングは 10.96 秒, Ate ペアリングは 17.93 秒という結果を示した. 宮崎等の実装は有限体 $\mathbb{F}_{2^{239}}$ 上の埋め込み次数 4 の η_T ペアリングを用い, 演算時間 1.93 秒という結果を示した.

最も高速なペアリングの一つに標数 $p = 3$ の超特異楕円曲線上の η_T ペアリングがある. 本稿では MICAz 上で TinyOS を用いて有限体 $\mathbb{F}_{3^m}$ 上の η_T ペアリングの実装を行った. 本稿の実装では, 同じ有限体 $\mathbb{F}_{3^m}$ 上の η_T ペアリングである石黒等 [8] と同等のセキュリティを実現するため, $m = 97$ を選択した. 本実装で用いた η_T ペアリングアルゴリズムは, 有限体 $\mathbb{F}_{3^{97}}$ の加算, 乗算, 3 乗算, 逆元算が必要となる. 初期実装では, nesC 言語を用い実装をした. 乗算に改良 Comb 法 [21], 3 乗算は石黒等により提案された事前計算と並び替えテーブルを用いた高速化方法 [8], 乗法逆元は拡張ユークリッド互除法を用いて実装した [11]. η_T ペアリングにおける, メインループと最終べきでの $\mathbb{F}_{3^{6m}}$ の乗算として Gorla 等の手法を用いた [5]. この結果, ペアリングの演算時間が 5.2 秒となった. 初期実装におけるペアリング演算での有限体の演算時間の割合は, 加算, 乗算, 3 乗算, 逆元算がそれぞれ 3.3%, 90.2%, 5.4%, 1.1%である. 従って, ペアリングの高速化には, 最も演算時間の割合が大きい有限体の乗算の高速化が重要であった.

本稿では有限体乗算の高速化は, 乗算の構成要素である加算の高速化により行った. 有限体 $\mathbb{F}_{3^{97}}$ の加算の高速化は川原等によって提案された最小論理命令数での有限体の加算 [12] を用い, nesC 言語を用いて実装をした. その結果, 初期実装の加算に比べ, 約 1.2 倍の高速化となり, 有限体の乗算も初期実装に比べ, 約 1.2 倍の高速化となった. 上記の高速化の結果, ペアリングの演算時間は 3.9 秒となった.

2 η_T ペアリングの実装方法

本章では有限体 $\mathbb{F}_{3^m}$ 上の η_T ペアリング, パラメータ選択, 有限体の表現方法, 及び本稿の初期実装について説明する.

2.1 η_T ペアリングの定義

η_T ペアリングは標数 3 の場合, 超特異楕円曲線 $y^2 = x^3 - x + b, b = \pm 1$ で定義される. 有限体 $\mathbb{F}_{3^m}$ で定義された超特異楕円曲線の点集合は次の通りである.

$$E(\mathbb{F}_{3^m}) = \left\{(x, y) \in (\mathbb{F}_{3^m})^2 | y^2 = x^3 - x + b\right\} \cup \{\infty\}$$

ここで ∞ は無限遠点である. r を $r \mid \#E(\mathbb{F}_{3^m})$ となる最大の素数とすると, $r \mid (3^{6m} - 1)$ を満たすことが知られている. $E(\mathbb{F}_{3^m})$ の位数 r の部分群を $E(\mathbb{F}_{3^m})[r]$ とすると, η_T ペアリングは以下のように定義される.

$$\begin{aligned}\eta_T : E(\mathbb{F}_{3^m})[r] \times E(\mathbb{F}_{3^{6m}})/rE(\mathbb{F}_{3^{6m}}) \\ \rightarrow \mathbb{F}^*_{3^{6m}}/(\mathbb{F}^*_{3^{6m}})^r\end{aligned}$$

$E(\mathbb{F}_{3^{6m}})/rE(\mathbb{F}_{3^{6m}})$ の元は, 点 $Q = (x, y) \in E(\mathbb{F}_{3^m})$ をリフティングする distortion 写像 $\phi(x, y) = (-x + \rho, y\sigma)$ を利用して $E(\mathbb{F}_{3^m})$ の元から生成できる. つまり, η_T ペアリングの入力は $E(\mathbb{F}_{3^m})$ から元を二つ選ぶことが可能である. η_T ペアリングは $P, Q \in E(\mathbb{F}_{3^m})[r], a \in \mathbb{Z}$ に対し, 双線形性 $\eta_T(aP, Q) = \eta_T(P, aQ) = \eta_T(P, Q)^a$ を満たす.

2.2 パラメータ選択

η_T ペアリングの安全性は, 楕円曲線上の離散対数問題の困難性および有限体上の離散対数問題の困難性に基づいている. 従って, 求められるセキュリティレベルに応じた大きな素数を持つ $\mathbb{F}_{3^m}$ の次数 m と曲線を選択する必要がある. $\mathbb{F}_{3^m}$ 上の η_T ペアリングのパラメータとして, 拡大次数 m と超特異楕円曲線の係数 b がある.

本稿では石黒等の $\mathbb{F}_{3^{97}}$ 上の η_T ペアリング実装 (埋め込み次数 6)[8], 宮崎等の $\mathbb{F}_{2^{239}}$ 上の η_T ペアリング実装 (埋め込み次数 4)[14] と比較する為に, $m = 97, b = 1$ を選択した.

2.3 η_T ペアリングのアルゴリズム

本節では η_T ペアリングのアルゴリズムについて説明する. η_T ペアリングは Duursma-Lee アルゴリズム [4] のループ回数を約半分に減少させたアルゴリズムである [1]. η_T ペアリングはメインループと最終冪の 2 つの処理により構成される.

メインループは η_T ペアリングの改良されたアルゴリズムとして, Beuchat 等によって従来のループの 2 つを 1 つにまとめ, ループの回数を半分にし高速化したアルゴリズムを用いた [3]. そのアルゴリズムを Algorithm 1 に示す. 最終冪は白勢等によって

トーラス T_2 を用いることで高速化されたアルゴリズムを用いた [19].

η_T ペアリングにおいて, 有限体の乗算の回数は, 拡大次数 m に比例し, また有限体の乗算の演算時間は m^2 に比例する. 従って, η_T ペアリングの演算時間は m^3 に比例する.

Algorithm 1 Refined η_T pairing [21]

INPUT: $P=(x_p,y_p), Q=(x_q,y_q)$
OUTPUT: $\eta_T(P,Q)\in \mathbb{F}^*_{3^{6m}}$

1: $y_p \leftarrow -y_p, d \leftarrow 1$
2: $f \leftarrow -y_p(x_p+x_q+1)+y_q\sigma+y_p\rho$
3: $u \leftarrow x_p+x_q+d$
4: $g \leftarrow y_py_p\sigma-u^2-u\rho-\rho^2$
5: $f \leftarrow fg$
6: $y_p \leftarrow -y_p, x_q \leftarrow x_q^9, y_q \leftarrow y_q^9$
7: $d \leftarrow d-1,\ f \leftarrow f^3$
8: **for** $i=0$ to $\frac{m-1}{4}-1$ **do**
9: $\quad u \leftarrow x_p+x_q+d$
10: $\quad g_1 \leftarrow (y_py_p\sigma-u^2-u\rho-\rho^2)^3$
11: $\quad y_p \leftarrow -y_p, x_q \leftarrow x_q^9, y_q \leftarrow y_q^9$
12: $\quad u \leftarrow x_p+x_q+d-1$
13: $\quad g_2 \leftarrow y_py_p\sigma-u^2-u\rho-\rho^2$
14: $\quad y_p \leftarrow -y_p, x_q \leftarrow x_q^9, y_q \leftarrow y_q^9$
15: $\quad d \leftarrow d+1$
16: $\quad g \leftarrow g_1g_2$
17: $\quad f \leftarrow (f^3g)^3$
18: **end for**
19: **return** f

2.4 有限体 $\mathbb{F}_{3^m}$ の元の表現

素体 $\mathbb{F}_3=\{0,1,2\}$ は要素数が 3 であるため, 1 ビットでは表すことができない. そこで hi ビット, lo ビットの 2 ビットで $\mathbb{F}_3$ の元を表現する方法が Harrison 等によって提案されている [7]. $a\in\mathbb{F}_3$ として a の hi ビットを a_{hi}, a の lo ビットを a_{lo} と表現する. このとき元 a は $a=(a_{hi},a_{lo})$ と表せ, $\mathbb{F}_3$ の元を以下のように対応させることができる.

$$0\mapsto(0,0),\ 1\mapsto(0,1),\ 2\mapsto(1,0)$$

有限体 $\mathbb{F}_{3^m}$ の元 $A(x)$ は $m-1$ 次の多項式として表現される. つまり, m 個の $\mathbb{F}_3$ の元により表現されるため, m 個の a_{hi} と a_{lo} が必要になる. ワード長を W とすると, $\lceil\frac{m}{W}\rceil$ 個の a_{hi} と a_{lo} が必要になる. 今回使用した MICAz のワード長は 8 ビットであるため, 8 個の hi ビット, または 8 個の lo ビットをまとめて 1 ワードに記憶する. $W=8$, $m=97$ より, $\lceil\frac{97}{8}\rceil=13$ となり, hi ビットと lo ビットのために各々13 ワード必要になる. 配列の j 番目の要素は $A(x)$ の hi ビットと lo ビットをまとめて $A[j]$ と表すと次のように $\mathbb{F}_{3^{97}}$ の元を表現できる.

$$A[12]=(0,0,\cdots,0,a_{96}),A[11]=(a_{95},a_{94},\cdots,a_{88}),$$
$$\cdots,A[1]=(a_{15},a_{14},\cdots,a_8),A[0]=(a_7,a_6,\cdots,a_0)$$

$A[12]$ の左 7 ビットは 97 次以上の係数が存在しないため 0 を詰める. また, $A[j]$ の k 番目のビットを $A[j]_k$ と表す. 以下に本稿で使用する記号を定義する.

W:ワード長
N:有限体を格納する配列の要素数=$\lceil\frac{m}{W}\rceil$
$A\&B$：論理積
$A\,|\,B$：論理和
$A\wedge B$：排他的論理和
$A[j]_k$：配列 A の j 番目の要素の k ビット目

2.5 有限体 $\mathbb{F}_{3^m}$ の演算

係数が基礎体 $\mathbb{F}_3$ の元である多項式の集合を $\mathbb{F}_3[x]$ と表す. $f(x)$ を m 次のモニック既約多項式とすると有限体 $\mathbb{F}_{3^m}$ は $\mathbb{F}_3[x]/f(x)$ として表される. 有限体 $A(x)\in\mathbb{F}_{3^m}$ は, 多項式表現では

$$A(x)=a_{m-1}x^{m-1}+a_{m-2}x^{m-2}+\cdots+a_0x^0$$

と表すことができる. ここで $a_i\in\mathbb{F}_3,(i=0,1,2,\cdots,m-1)$ である.

$A(x),B(x)\in\mathbb{F}_{3^m}$ に対する各有限体演算は以下の方法を用いた.

加算の実装では, Harrison らによって提案された手法を用いた [7]. その手法とは, 前節で説明したビット表現を用いることにより 1 回の論理積, 4 回の論理和, 2 回の排他的論理和の計 7 回の論理命令で 1 ワードの加算を実現する. そのアルゴリズムを Algorithm 2 に示す.

Algorithm 2 Addition in GF(3^m)[7]

INPUT: $a=(a_{hi},a_{lo}),b=(b_{hi},b_{lo})\in\mathbb{F}_{3^m}$
OUTPUT: $c=(c_{hi},c_{lo})=a+b\in\mathbb{F}_{3^m}$

1: $t \leftarrow (a_{hi}\,|\,a_{lo})\,\&\,(b_{hi}\,|\,b_{lo})$
2: $c_{hi} \leftarrow t\wedge(a_{hi}\,|\,b_{hi})$
3: $c_{lo} \leftarrow t\wedge(a_{lo}\,|\,b_{lo})$

乗算は, shift_additon 法, window 法, LR_Comb 法 [6], と改良 Comb 法 [21] を実装した. 本稿では, これらの中で最速であった改良 Comb 法を選択した. 改良 Comb 法はシフトの回数が $W-1$ 回となる. $W=8$ ビットである MICAz では高々7 回のシフト処理で良いため, shift_additon 法より高速することが可能である. そのアルゴリズムを Algorithm 3 に示す.

Algorithm 3 Multiplication in GF(3^m)[21]

INPUT: $A(x), B(x) \in \mathbb{F}_{3^m}$
OUTPUT: $C = (x) = A(x) \cdot B(x)$
1: $C \leftarrow 0$
2: **for** $j \leftarrow 0$ to $N-1$ **do**
3: $\quad C(x) \leftarrow C(x) + A[j] \cdot B(x)x^{jw}$
4: **end for**
5: **for** $i \leftarrow 1$ to $W-1$ **do**
6: $\quad$ **for** $i \leftarrow 1$ to $W-1$ **do**
7: $\quad\quad C(x) \leftarrow C(x) + A[j]_i \cdot B(x)x^{jw+1}$
8: $\quad$ **end for**
9: **end for**
10: **return** $C(x)$

3 乗算は石黒等により提案された事前計算と並び替えテーブルを用いた高速化方法を用いた [8]. また, 乗法逆元は拡張ユークリッド互除法を用いて実装した [11].

2.6 既約多項式の選択について

有限体乗算は多項式乗算とリダクションの 2 つの処理に分かれている. 乗算の高速化において, 多項式乗算だけではなく, リダクション処理を速くすることも重要になる. 中島等により, 既約多項式を $f(x) = x^m + ax^k + b$ とした場合, k がワード長 W の整数倍となるような既約多項式を選択することによって乗算におけるリダクションを高速に行えることが示された [15]. MICAz のマイクロコントローラ ATmega128L のワード長は 8 ビットである. 本稿では, 既約多項式 $f(x) = x^{97} + x^{16} + 2$ を選択した.

2.7 拡大体 $\mathbb{F}_{3^{3m}}$, $\mathbb{F}_{3^{6m}}$ の演算

本稿では $\mathbb{F}_{3^m}$ の 3 次拡大 $\mathbb{F}_{3^{3m}}$ を 2 次拡大することにより, 6 次拡大体 $\mathbb{F}_{3^{6m}}$ を構成する.

3 次拡大体の多項式表現は, $a_0, a_1, a_2 \in \mathbb{F}_{3^m}$ に対して, $A(\rho) = a_2\rho^2 + a_1\rho + a_0$ となる. 3 次拡大の既約多項式は $g(\rho) = \rho^3 - \rho - 1$ とした. また, 2 次拡大体の多項式表現は, $\alpha_0, \alpha_1 \in \mathbb{F}_{3^{3m}}$ に対して, $A(\sigma) = \alpha_1\sigma + \alpha_0$ となる. 2 次拡大の既約多項式は $g(\sigma) = \sigma^2 + 1$ とした. 6 次拡大体 $\mathbb{F}_{3^{6m}}$ の元 A は $\alpha_j \in \mathbb{F}_{3^{3m}}$, $(j = 0, 1)$, $a_i \in \mathbb{F}_{3^m}$, $(i = 0, 1, \cdots, 5)$ とすると,

$$\begin{aligned} A &= \alpha_1\sigma + \alpha_0 \\ &= a_5\sigma\rho^2 + a_4\rho^2 + a_3\sigma\rho + a_2\rho + a_1\sigma + a_0 \\ &= (a_5, a_4, a_3, a_2, a_1, a_0) \end{aligned}$$

と表す.

$\mathbb{F}_{3^{3m}}$, $\mathbb{F}_{3^{6m}}$ の加算, 減算, 3 乗算, 乗法逆元の演算は石黒等 [8] と同様の演算を行っている.

2.8 η_T ペアリングの実装

η_T ペアリングの初期実装では, $\mathbb{F}_{3^{97}}$ 上の η_T ペアリングを MICAz 上に nesC 言語を用いて実装を行った. ここで, η_T ペアリングの初期実装における, 有限体の演算アルゴリズムは 2.5 節で説明したものを用いた.

初期実装において有限体演算の割合は表 1 の結果となった. それらの演算を用いた実装の結果, η_T ペアリングの演算時間は 5.2 秒となった. この結果から, η_T ペアリングを高速化するためには, 有限体演算の中で割合が 90.2%と最も多い乗算を高速化することが効果的である.

表 1: 初期実装での η_T ペアリングの演算時間の割合

	加算	乗算	3 乗算	乗法逆元
割合	3.3%	90.2%	5.4%	1.1%

3 高速化手法

本章では, 有限体 $\mathbb{F}_{3^m}$ 上の加算の高速化方法, 乗算の高速化見積もりについて説明する.

3.1 高速化概要

2.8 節より, η_T ペアリングの高速化には, 有限体 $\mathbb{F}_{3^m}$ の乗算を高速化することが効果的である. 初期実装では, 有限体の乗算を改良 Comb 法を用いて実装した [21]. 改良 Comb 法とは, $B(x)$ を最上位ビットからワード長おきに走査し $A(x)$ の加算を行うことにより演算を行う. つまり乗算は加算を繰り返すことによって構成されており, 加算の高速化が乗算の高速化に繋がる. 従って, 乗算の高速化には, 加算の高速化が重要である.

3.2 有限体 $\mathbb{F}_{3^m}$ の加算の高速化

本節では有限体 $\mathbb{F}_{3^m}$ の加算の高速化手法について説明する. 有限体の加算の高速化は, 川原等によって提案された最小論理命令数での有限体の加算を用いることにより行った [12].

初期実装では, Harrison 等が提案した 7 回の論理命令を用いて加算を行っていた [7]. これに対して, 川原等の方式では, 有限体 $\mathbb{F}_3$ の元の 2 ビット表現を変更し, 6 回の論理命令での加算を実現する. 川原の方式を 2.4 節で説明したビット表現を用いると表 2 のように表現される. この 2 ビット表現を利用することにより, 2 回の論理和と, 4 回の排他的論理和で

加算を構成できる. アルゴリズムを Algorithm 4 に示す. 高速化により, 初期実装と比較して加算の演算時間を約 1.2 倍高速化することができた.

表 2: $\mathbb{F}_3$ の元の表現の比較

従来方式 [7]		
a	a_{hi}	a_{lo}
0	0	0
1	0	1
2	1	0

川原等 [12]		
a	a_{hi}	a_{lo}
0	1	1
1	1	0
2	0	1

Algorithm 4 Addition in GF(3^m)[12]

INPUT: $a = (a_{hi}, a_{lo}), b = (b_{hi}, b_{lo}) \in \mathbb{F}_{3^m}$
OUTPUT: $c = (c_{hi}, c_{lo}) = a + b \in \mathbb{F}_{3^m}$
1: $s \leftarrow a_{hi} \wedge b_{hi}$
2: $t \leftarrow a_{lo} \wedge b_{lo}$
3: $c_{hi} \leftarrow t \mid (s \wedge a_{lo})$
4: $c_{lo} \leftarrow s \mid (t \wedge a_{hi})$

3.3 有限体の乗算の高速化

有限体乗算の高速化手法として, 3.2 節の加算を用い改良 Comb 法の実装を行った. 加算の高速化手法として有限体 $\mathbb{F}_3$ の元の表現を変えたことにより, シフト処理も変える必要がある. 元 $0 \in \mathbb{F}_3$ は $(1, 1)$ に対応するため, シフト後に代入する値を 0 から 1 に変換する必要がある. この変換は, 下位 $i-1$ ビットが 0, かつ他のビットが 1 であるマスク M と a_{hi}, a_{lo} に対して AND 演算を行うことで計算できる. これにより, シフト演算において余分な計算コストが必要となる. しかしながら, 改良 Comb 法では $\mathbb{F}_3$ の元のシフトの回数が $W-1$ 回であり, $W = 8$ ビットである MICAz では高々7 回のシフト処理で良いため, 有限体乗算の演算速度に大きな影響を及ぼすことはなかった.

4 実装結果

本章では, ペアリング暗号の初期実装と提案方式との演算時間との比較, 既存の η_T ペアリング実装との比較について説明をする.

4.1 初期実装と提案方式の比較

初期実装と提案方式を有限体 $\mathbb{F}_{3^m}$ の演算, η_T ペアリングの処理速度から比較した. その結果を表 3 に示す.

初めに, 本実装の初期実装と提案方式の比較を行う. 有限体の加算は, 川原等の方式 [12] を用いたことにより, 約 17%高速な結果となった. 加算の高速化の結果, 有限体の乗算も約 16%高速な結果となった. 加算の高速化の際に $\mathbb{F}_3$ の元の表現を変更したことにより, シフト処理で余分なコストが生じた. これに伴い, 有限体の 3 乗算, 逆元算の演算時間が初期実装と比較して若干遅くなってしまった. しかしながら, 3 乗算, 逆元算の演算時間は η_T ペアリングの構成において割合が小さいため, ペアリングの演算時間に大きな影響を及ぼすことはなかった. 初期実装での有限体の各種演算での η_T ペアリングの演算時間は 5.2 秒であった. 加算, 乗算の高速化の結果, η_T ペアリングの演算時間は 3.9 秒と約 25%の高速化となった.

表 3: 有限体 $\mathbb{F}_{3^m}$ と η_T ペアリングの計算時間

	初期実装 (2 節)	提案方式 (3 節)
加算	0.047ms	0.039ms
乗算	5.5ms	4.7ms
3 乗算	0.45ms	0.66ms
逆元算	95.9ms	98.1ms
メインループ	4,158ms	3,164ms
最終べき	757ms	738ms
ペアリング	5,157ms	3,892ms

4.2 既存のペアリング暗号実装との比較

本節では, ATmega128L 上で実装されたペアリング演算との演算時間とメモリ使用量の比較を行う. 比較対象は本実装と Oliveria 等の TinyPBC[17], 石黒等の η_T ペアリング実装 [8], Szczechowiak 等による NanoECC[20], 宮崎等の η_T ペアリング実装 [8] とした. その結果を表 4 に示す.

初めに, 演算時間について比較を行う. 本実装と石黒等 [8], 宮崎等 [14] の実装は同等のセキュリティである. しかしながら, 拡大次数が $m = 271$ である TinyPBC[17], NanoECC[20] とはセキュリティが異なっている. 従って, 本実装と比較を行う為に, 2.3 節で説明したように η_T ペアリングの演算時間は有限体の拡大次数 m を用いて m^3 に比例することを利用して, 同等のセキュリティとしてペアリングの演算時間を見積もる. 見積もりの結果, 初期実装では 7.17 秒, 提案方式では 5.41 秒となった. まず, 提案方式と有限体 $\mathbb{F}_{2^{239}}$ を用いた η_T ペアリングと比較を行った. 提案方式は, TinyPBC[17] とほぼ同等の速度となり, NanoECC[20] よりも約 2 倍高速となった. 次に, 同等のセキュリティである石黒等 [8], 宮崎

表 4: ATmega128L 上でのペアリング暗号実装の比較

	石黒等 [8]	TinyPBC[17]	NanoECC[20]	宮崎等 [14]	初期実装	提案方式
言語	nesC	nesC	nesC	nesC, asm	nesC	nesC
有限体	$\mathbb{F}_{3^{97}}$	$\mathbb{F}_{2^{271}}$	$\mathbb{F}_{2^{271}}$	$\mathbb{F}_{2^{239}}$	$\mathbb{F}_{3^{97}}$	$\mathbb{F}_{3^{97}}$
計算時間 (sec)	5.79	5.45	10.96	1.93	5.16	3.89
ROM(byte)	17,284	47,948	53,500	35,012	19,712	20,878
RAM(byte)	628	368	2,800	1,231	704	827

等 [14] と提案方式の比較を行った. 有限体 $\mathbb{F}_{2^{239}}$ を用いた宮崎等の実装と比較すると, 約 2 倍も遅い. しかしながら, 同じ有限体 $\mathbb{F}_{3^{97}}$ を用いた石黒等の実装よりも提案方式は約 1.5 倍高速な結果となった.

次に, メモリ使用量について比較を行う. 提案方式の ROM の使用量は石黒等 [8] よりも 1.2 倍多かった. TinyPBC[17] より約 2.3 倍, NanoECC[20] より約 2.7 倍, 宮崎等 [14] より約 1.7 倍少ない結果となった. また, 提案方式の RAM の使用量は TinyPBC[17] より約 2.2 倍, 石黒等 [8] よりも 1.3 倍多かった. NanoECC[20] より約 3.4 倍, 宮崎等 [14] よりも約 1.5 倍少ない結果となった.

5 まとめ

本稿では, 標数 3 の有限体 $\mathbb{F}_{3^{97}}$ 上の超特異曲線を用いた η_T ペアリングを MICAz 上に実装し, 処理速度の評価を行った. 実装は MICAz 上に, TinyOS を用いて nesC 言語でプログラムを作成した. 高速化にあたり有限体の加算を高速化した. その結果, 初期実装では 5.2 秒であった MICAz 上での η_T ペアリングの計算時間が 3.9 秒まで改善された. これにより, 同等のパラメータである有限体 $\mathbb{F}_{3^{97}}$ を用いている石黒等の実装よりも約 1.5 倍の高速化を実現することができた.

参考文献

[1] P. Baretto, S. Galbraith, C. Ó hÉigeartaigh , and M. Scott, "Efficient Pairing Computation on Supersingular Abelian Varieties", CRYPTO 2002, LNCS 2442, pp.354-369, 2002.

[2] J.-L. Beuchat, M. Shirase, T. Takagi, and E. Okamoto, "An algorithm for the η_T pairing calculation in characteristic three and its hardware implementation", *18th IEEE International Symposium on Computer Arithmetic, ARITH-18*, pp.97-104, 2007.

[3] J.-L. Beuchat, M. Shirase, T. Takagi, and E. Okamoto, "A Refined Algorithm for the η_T Pairing Calculation in Characteristic Three", Cryptology ePrint Archive Report 2007/311, 2007.

[4] I. Duursma , and H. Lee, "Tate pairing implementation for hyperelliptic curve $y^2 = x^p - x + d$", ASIACRYPT2003, LNCS 2894, pp. 111-123, 2003.

[5] E. Gorla, C. Puttmann, and J. Shokrollahi, "Explicit Formulas for Efficient Multiplication in $\mathbb{F}_{3^{6m}}$", The 14th Annual Workshop on Selected Areas in Cryptography, SAC 2007, to appear.

[6] D. Hankerson, A. Menezes, and S. Vanstone, Guide to Elliptic Curve Cryptography, Springer, 2004.

[7] K. Harrison, D. Page, and N.P. Smart, "Software implementation of finite fields of characteristic three, for use in pairing-based cryptosystems", LMS Journal of Computation and Mathematics, vol. 5, pp. 181-193, 2002.

[8] 石黒司, 白勢政明, 高木剛, "ATmega128L 上でのペアリング暗号の高速実装 ", 情報処理学会, コンピュータセキュリティシンポジウム, pp. 187-192, CSS 2007.

[9] IT 戦略本部, " IT 新改革－いつでも, どこでも, 誰でも IT の恩恵を実感できる社会の実現－ ", 2006

[10] IT 戦略本部, " 重点計画－ 2008", 2008

[11] Y. Kawahara, T. Takagi, and E. Okamoto, "Efficient Implementation of Tate Pairing on a Mobile Phone using Java", Computational Intelligence and Security, CIS 2006, Lecture Notes in Artificial Intelligence, Vol.4456, PP.396-405, 2007.

[12] Y. Kawahara, K. Aoki and T. Takagi, "Faster implementation of η_T pairing over GF(3^m) using minimum number of logical instructions for GF(3) Addition", Pairing 2008, Lecture Notes in Computer Science, Volume 5209, pp.289-296,2008.

[13] MICAz Hardware Description Available at "`http://www.xbow.jp/`".

[14] 宮崎行規, 白勢政明, 高木剛, "MICAz 上での高速ペアリング暗号実装 ", 情報処理学会, コンピュータセキュリティシンポジウム, pp. 199-2004, CSS 2008.

[15] T. Nakajima, T. Izu and T. Takagi, "Reduction Optimal Trinomials for Efficient Software Implementation of the η_T Pairing ", 2nd International Workshop on Security, IWSEC 2007, LNCS 4752, pp.44-57, 2007.

[16] L. Oliveira, D. Aranha, E. Morais, F. Daguano, J. López, and R, Dahab, "TinyTate: Identity-Based Encryption for Sensor Networks", Cryptology ePrint Archive, Report 2007/020, 2007.

[17] L. Oliveira, D. Aranha, E. Morais, F. Daguano, J. López, and R, Dahab, "TinyPBC: Pairings for Authenticated Identity-Based Non-Interactive Key Distribution in Sensor Networks", Cryptology ePrint Archive, Report 2007/482, 2007.

[18] A. Perrig, J. Stankovic, and D. Wagner, "Security in wireless sensor networks", Communications of the ACM, Vol.47 No.6, 2004.

[19] M. Shirase, T. Takagi , and E. Okamoto, "Some Efficient Algorithms for the Final Exponentiation of η_T Pairing", IPSEC 2007, LNCS, vol. 4464, pp. 254-268, 2007.

[20] P. Szczechowiak, L. Oliveira, M. Scott, M. Collier, and R. Dahab, "NanoECC:Testing the Limits of Elliptic Curve Cryptography in Sensor Network", EWSN 2008, LNCS 4913, pp.305-320, 2008.

[21] M. Yoshitomi, T. Takagi, S. Kiyomoto , and T. Tanaka, "Efficient Implementation of the Pairing on Mobilephones using BREW", WISA2007, LNCS 4867, pp.203-214, 2007.